Creating R packages

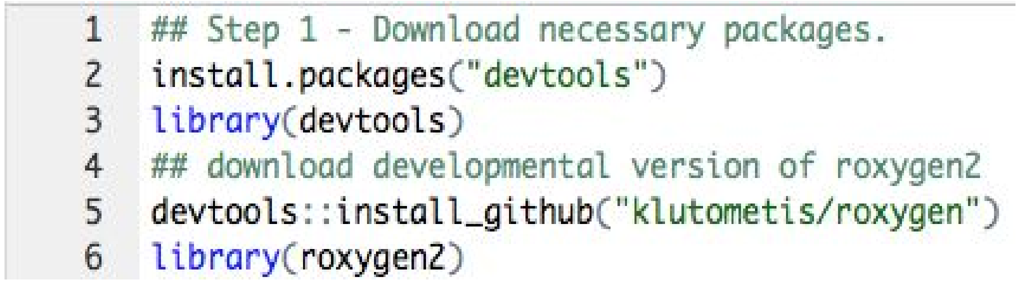
Khagay Nagdimov (khagay@nyu.edu)
mskilab (Dr. Marcin Imielinski)
December 8, 2016

Overview

A. Basic example
B. Complex example 1
C. Complex example 2
D. Questions

Step 1: Setup environment

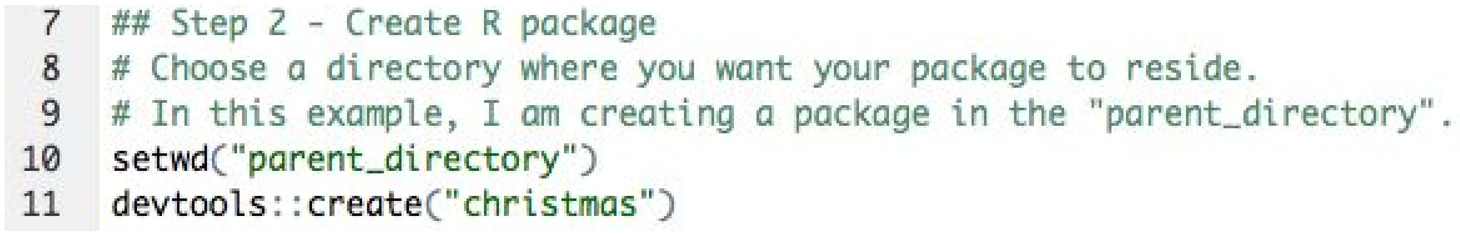
Install & Load devtools and roxygen2



```r
## Step 1 - Download necessary packages.
install.packages("devtools")
library(devtools)
## download developmental version of roxygen2
devtools::install_github("klutometis/roxygen")
library(roxygen2)
```

Step 2: Create an empty R package

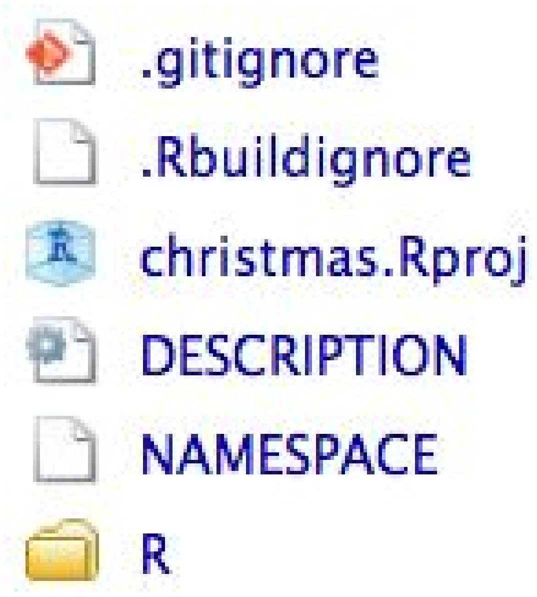
Create an empty package



```
## Step 2 - Create R package
# Choose a directory where you want your package to reside.
# In this example, I am creating a package in the "parent_directory".
setwd("parent_directory")
devtools::create("christmas")
```

Step 2 Continued

You should have this structure



```
.gitignore
.Rbuildignore
christmas.Rproj
DESCRIPTION
NAMESPACE
R
```

Background Info

Every R package has the following directories:

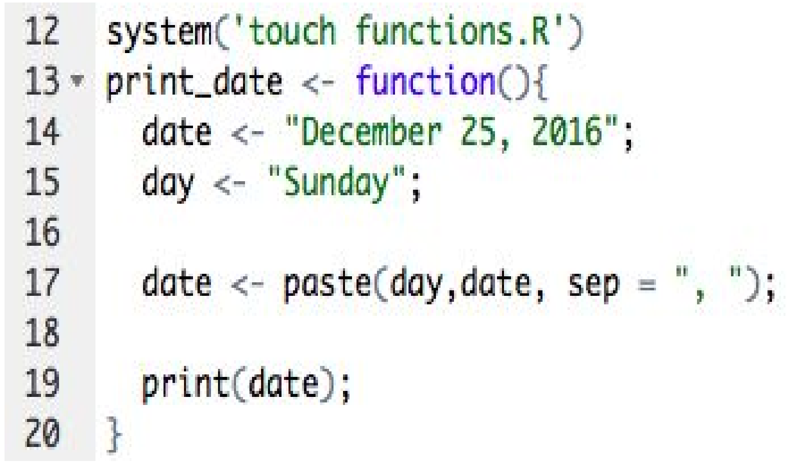
1. Man (has “R Documentation” a.k.a. Rd files that stores documentation for functions. Rd files syntax is similar to LaTeX)
2. R (houses the R code)

And the following files:

1. Description (lists the metadata - contact information, name and authors of package, etc.
2. Namespace (lists functions available once package is installed & loaded)

Step 3. Create Functions and Save Them

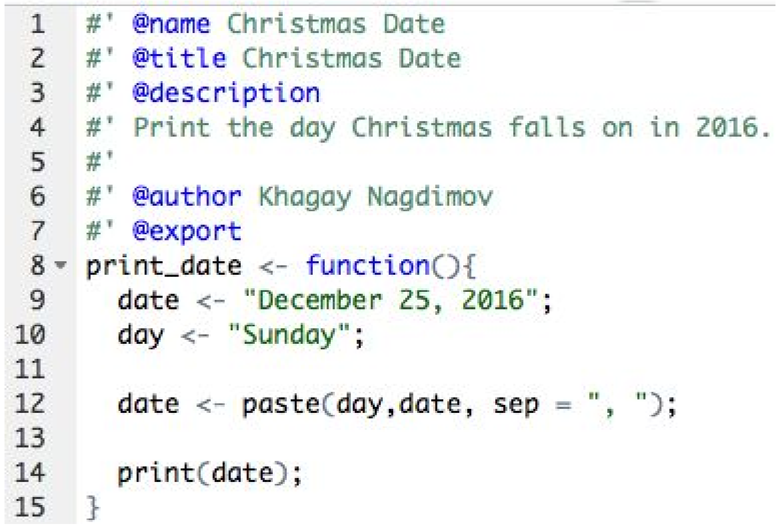
- I prefer to use Terminal and Emacs/ESS but, RStudio can do the job too.



```r
system('touch functions.R')
print_date <- function(){
  date <- "December 25, 2016";
  day <- "Sunday";

  date <- paste(day,date, sep = ", ");

  print(date);
}
```

Step 4. Add Documentation

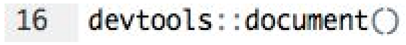
- Add tags (`#'`) that roxygen2 will catch and create appropriate documentation
- Add @export tag
- Add other tags depending on how extensive you want documentation to be.



```
#' @name Christmas Date
#' @title Christmas Date
#' @description
#' Print the day Christmas falls on in 2016.
#'
#' @author Khagay Nagdimov
#' @export
print_date <- function(){
  date <- "December 25, 2016";
  day <- "Sunday";

  date <- paste(day,date, sep = ", ");

  print(date);
}
```

Warnings:

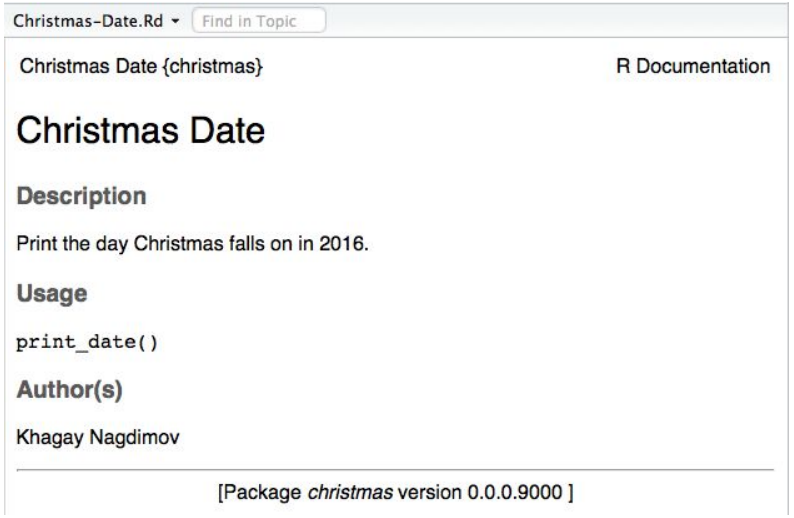
- Make sure there is no space between @export and the function.
- Run devtools::document() at root of package.



```
devtools::document()
```

Step 4: Add Documentation continued

Tags being translated ➜



Christmas Date {christmas} R Documentation

Christmas Date

Description

Print the day Christmas falls on in 2016.

Usage

```
print_date()
```

Author(s)

Khagay Nagdimov

[Package *christmas* version 0.0.0.9000]

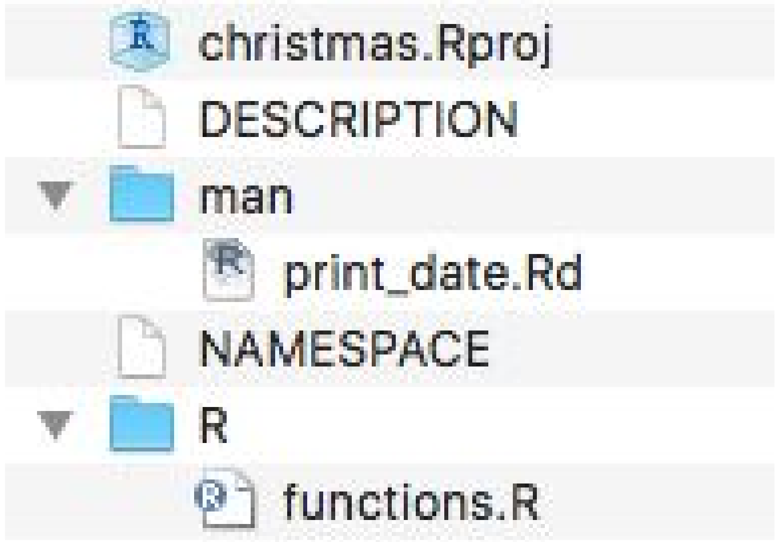
Without Devtools

Devtools is such a game changer because the document() function wraps the roxygen2::roxygenize() which generates the documentation for files as well as exporting functions/classes to the NAMESPACE file.

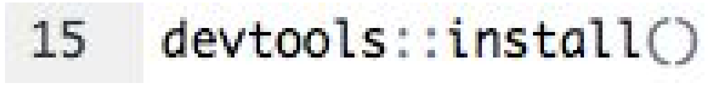
Alternative is to use utils::package_skeletion() and roxygen2::roxygenize() but, devtools wraps functions in one package and performs checks for you.

Test

You should have this structure


christmas.Rproj
DESCRIPTION
man
print_date.Rd
NAMESPACE
R
functions.R

Step 5/Final: Install!

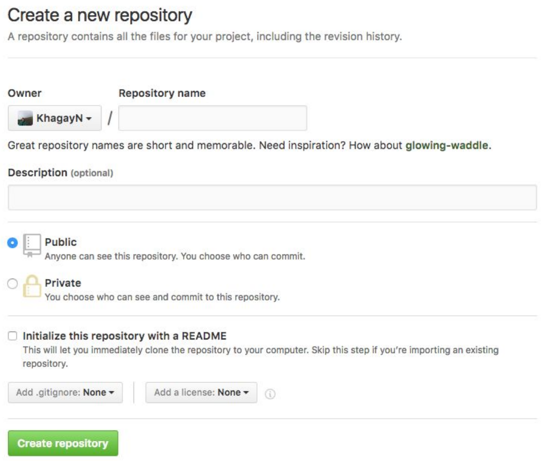


```
devtools::install()
```

Uses R CMD INSTALL to install the package. Also tries to install dependencies from CRAN.

Releasing Package off to the World 😢

Create Repository on GitHub (https://github.com/new)


Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: KhagayN / Repository name:

Great repository names are short and memorable. Need inspiration? How about glowing-waddle.

Description (optional)

[✓ Public] Anyone can see this repository. You choose who can commit.

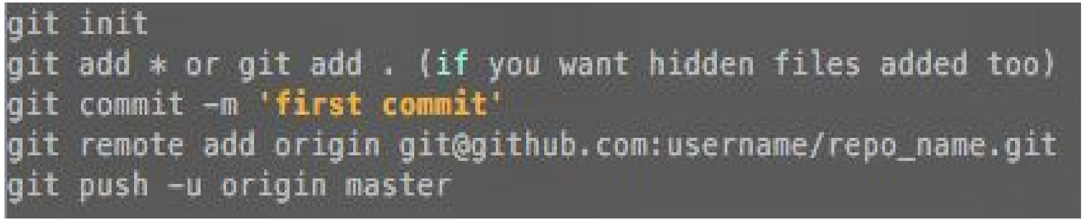
Private: You choose who can see and commit to this repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

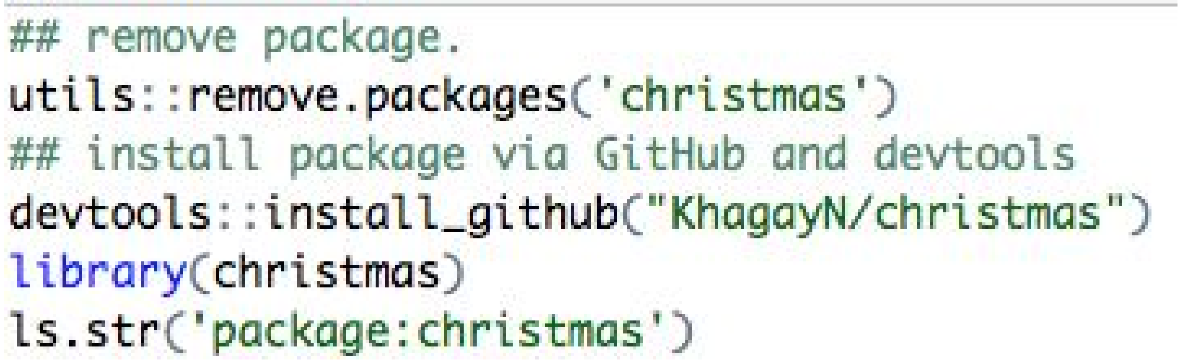
Add a license: None

Go to directory in Terminal (if Mac) and run:


```
git init
git add * or git add . (if you want hidden files added too)
git commit -m 'first commit'
git remote add origin git@github.com:username/repo_name.git
git push -u origin master
```

Releasing Package off to the World😢 Continued

Run the following:



```r
## remove package.
utils::remove.packages('christmas')
## install package via GitHub and devtools
devtools::install_github("KhagayN/christmas")
library(christmas)
ls.str('package:christmas')
```

Putting a package on CRAN

You have been forewarned!

To publish on CRAN follow guidelines here: https://cran.r-project.org/

Some restrictions:

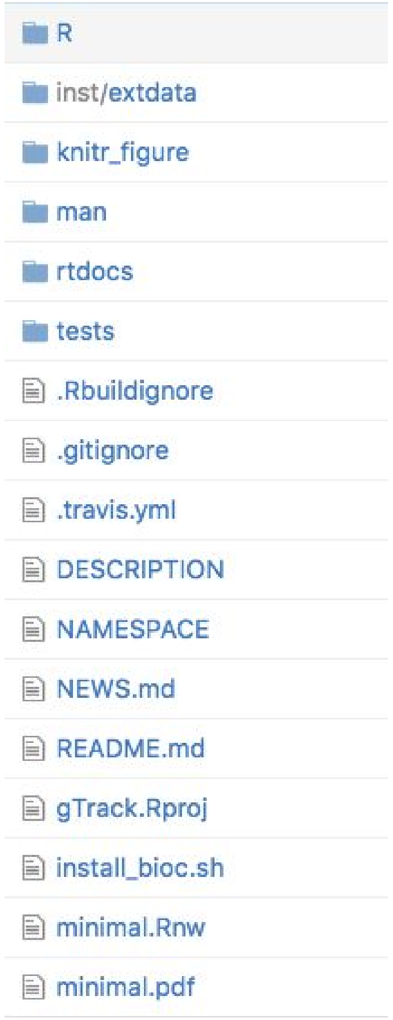
Package size must be 5MB

R CMD check must be run and no errors returned

Example

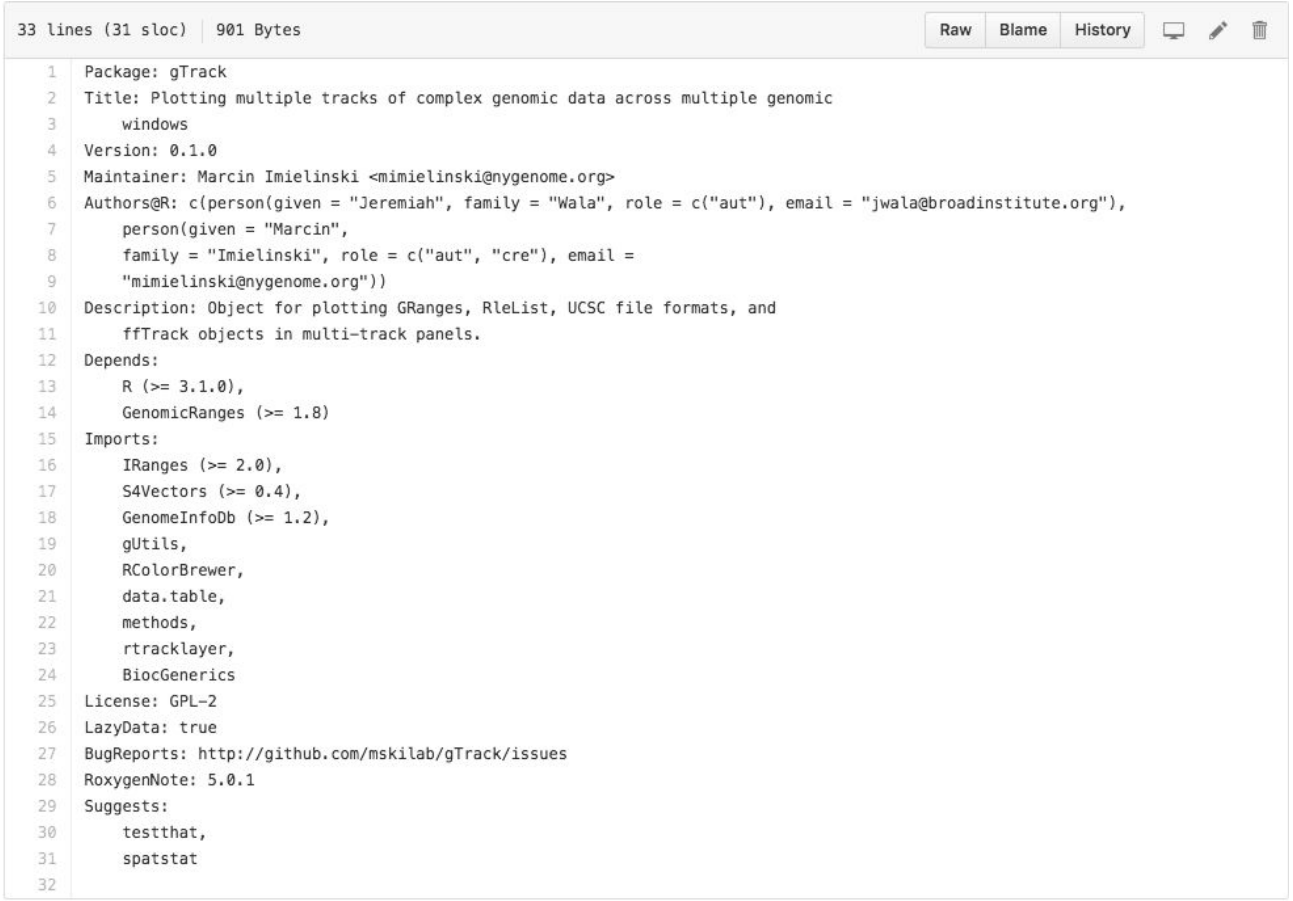
Imielinski’s gTrack:

https://github.com/mskilab/gTrack



R
inst/extdata
knitr_figure
man
rtdocs
tests
.Rbuildignore
.gitignore
.travis.yml
DESCRIPTION
NAMESPACE
NEWS.md
README.md
gTrack.Rproj
install_bioc.sh
minimal.Rnw
minimal.pdf

gTrack DESCRIPTION file



```
Package: gTrack
Title: Plotting multiple tracks of complex genomic data across multiple genomic
    windows
Version: 0.1.0
Maintainer: Marcin Imielinski <mimielinski@nygenome.org>
Authors@R: c(person(given = "Jeremiah", family = "Wala", role = c("aut"), email = "jwala@broadinstitute.org"),
    person(given = "Marcin",
    family = "Imielinski", role = c("aut", "cre"), email =
    "mimielinski@nygenome.org"))
Description: Object for plotting GRanges, RleList, UCSC file formats, and
    ffTrack objects in multi-track panels.
Depends:
    R (>= 3.1.0),
    GenomicRanges (>= 1.8)
Imports:
    IRanges (>= 2.0),
    S4Vectors (>= 0.4),
    GenomeInfoDb (>= 1.2),
    gUtils,
    RColorBrewer,
    data.table,
    methods,
    rtracklayer,
    BiocGenerics
License: GPL-2
LazyData: true
BugReports: http://github.com/mskilab/gTrack/issues
RoxygenNote: 5.0.1
Suggests:
    testthat,
    spatstat

```

Explanation of gTrack NAMESPACE

Depends: Packages that must be present to install this package.

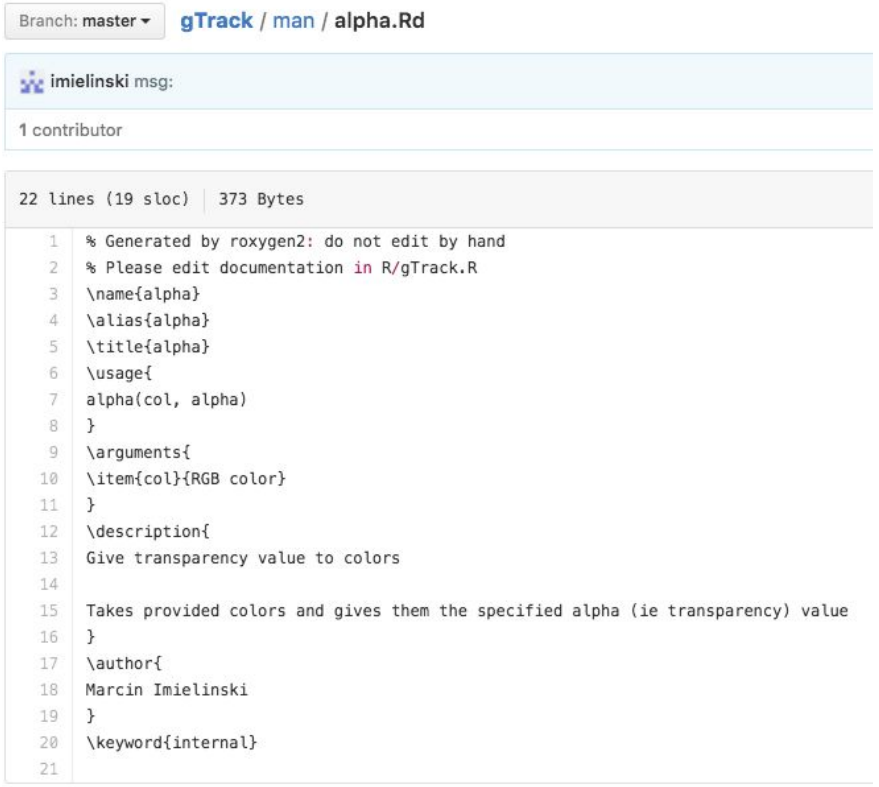
Imports: Safer than Depends because it limit conflicts with namespace. Packages in the Depends field, have all their functions loaded into the interpreter session while packages in the Imports, have their functions linked to a package namespace.

LazyData: specifies that data from the package will only be loaded when used.

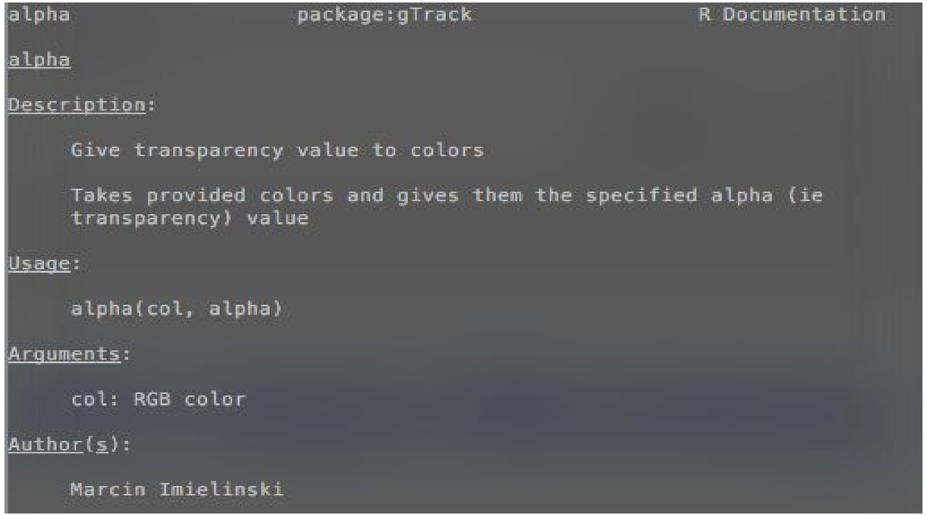
Suggests: Packages **useful** for this package but, **not necessary.**

Keywords: Only one useful is `@keywords internal`. Removes function from the documentation index.

Explanation of gTrack's man



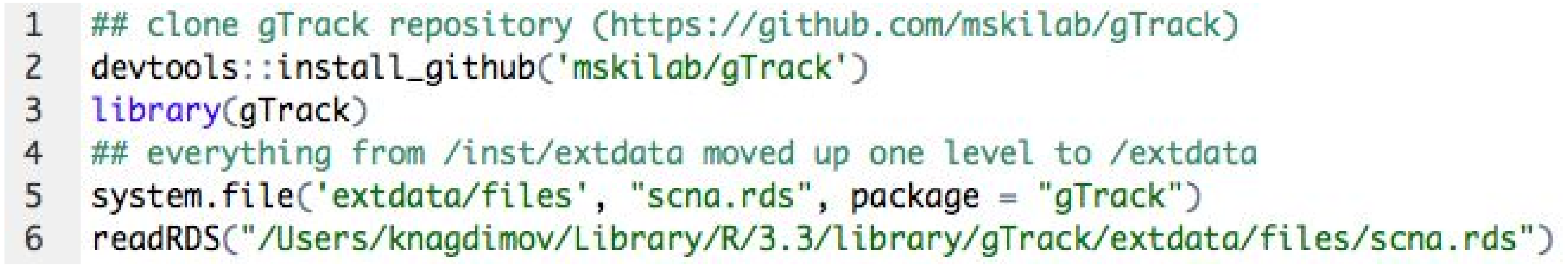
```
% Generated by roxygen2: do not edit by hand
% Please edit documentation in R/gTrack.R
\name{alpha}
\alias{alpha}
\title{alpha}
\usage{
alpha(col, alpha)
}
\arguments{
\item{col}{RGB color}
}
\description{
Give transparency value to colors

Takes provided colors and gives them the specified alpha (ie transparency) value
}
\author{
Marcin Imielinski
}
\keyword{internal}

```



```
alpha                  package:gTrack                  R Documentation

alpha

Description:

     Give transparency value to colors

     Takes provided colors and gives them the specified alpha (ie
     transparency) value

Usage:

     alpha(col, alpha)

Arguments:

     col: RGB color

Author(s):

     Marcin Imielinski
```

Explanation of gTrack's inst/extdata

- Including data with a package is a great feature!
- Save raw data in root_of_package/inst/extdata



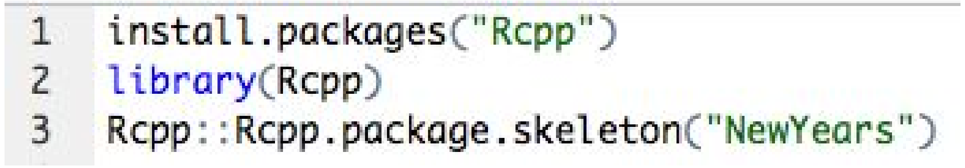
```r
## clone gTrack repository (https://github.com/mskilab/gTrack)
devtools::install_github('mskilab/gTrack')
library(gTrack)
## everything from /inst/extdata moved up one level to /extdata
system.file('extdata/files', "scna.rds", package = "gTrack")
readRDS("/Users/knagdimov/Library/R/3.3/library/gTrack/extdata/files/scna.rds")
```

Creating a trustworthy package

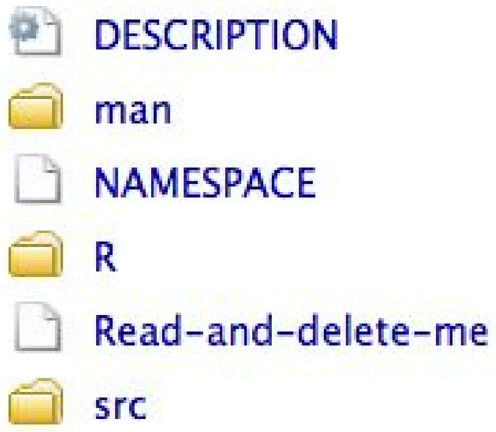
- Standards accepted by the community because they show the code is reproducible:
  - Travis (https://travis-ci.org/mskilab/gTrack)
  - Coveralls (https://coveralls.io/github/mskilab/gTrack?branch=master)
  - Read the docs (http://gtrack.readthedocs.io/en/latest/)

Complex Example Number 2 using Rcpp

R packages can incorporate C/C++ code!



```r
install.packages("Rcpp")
library(Rcpp)
Rcpp::Rcpp.package.skeleton("NewYears")
```

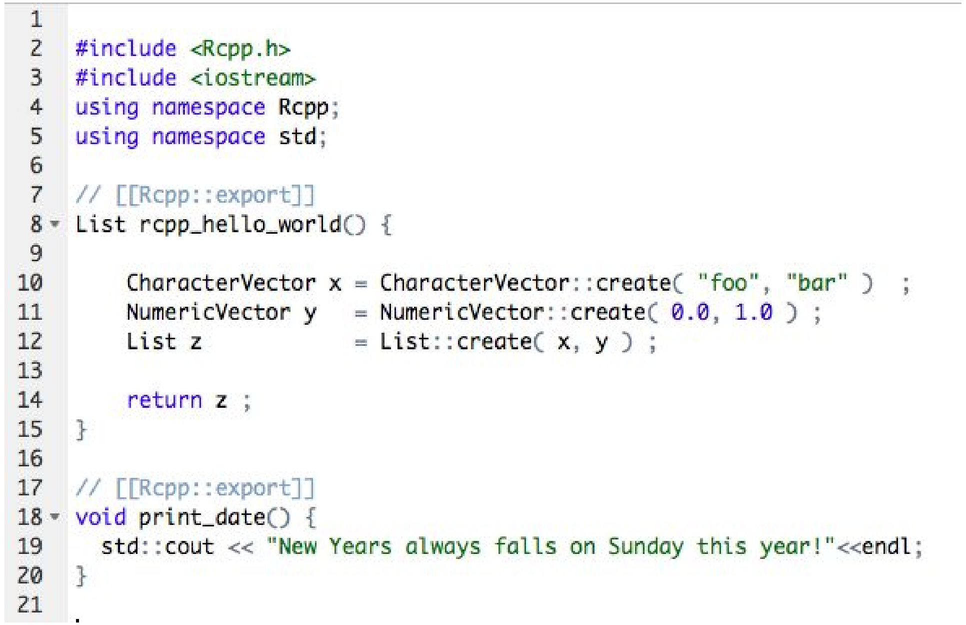


DESCRIPTION
man
NAMESPACE
R
Read-and-delete-me
src

Creating new functions

- Write C++ function that'll be translated into R.

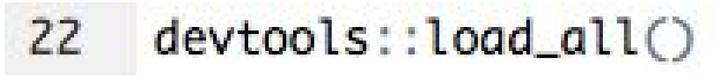
- Always include **//[[Rcpp::export]]** tag before function.



```
#include <Rcpp.h>
#include <iostream>
using namespace Rcpp;
using namespace std;

// [[Rcpp::export]]
List rcpp_hello_world() {

    CharacterVector x = CharacterVector::create( "foo", "bar" )  ;
    NumericVector y   = NumericVector::create( 0.0, 1.0 ) ;
    List z            = List::create( x, y ) ;

    return z ;
}

// [[Rcpp::export]]
void print_date() {
  std::cout << "New Years always falls on Sunday this year!"<<endl;
}
.
```

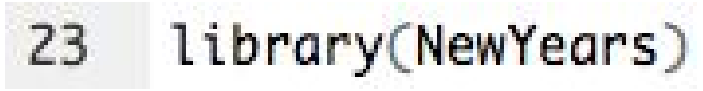
Install Package

- Install by running at the root of package:



```r
devtools::load_all()
```

- Load package:



```r
library(NewYears)
```

Publish

Publish via GitHub or CRAN depending on your goals.

Much thanks to:

Dr. Marcin Imielinski (mskilab)

http://r-pkgs.had.co.nz/description.html - Dr. Hadley Wickham (RStudio)

https://hilaryparker.com/2014/04/29/writing-an-r-package-from-scratch/ - Dr. Hillary Parker (Stich Fix)

Evan Biederstedt and Dr. Friederike Dundar (organizing and review)